BAB II

KAJIAN PUSTAKA

A. Kajian Teori

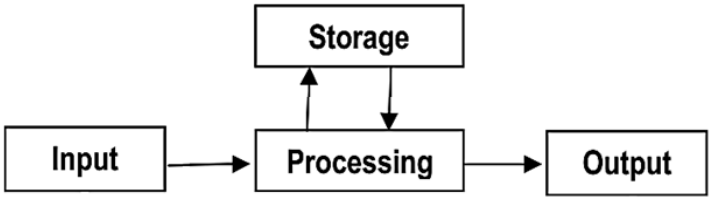
1. Data, Informasi, dan Sistem

Data merupakan representasi fakta dunia nyata yang mewakili suatu objek. Objek dapat berupa benda hidup maupun benda mati (Fathansyah, 2012: 2). Data berupa bentuk bahan mentah yang harus diolah untuk menjadi suatu informasi. Sumber dari informasi adalah data. Informasi adalah data yang diolah menjadi bentuk yang lebih berguna bagi penerima. Penerima kemudian menerima informasi tersebut untuk membuat suatu keputusan dan tindakan (Hartono, 2005: 8-9). Gibson, Ivancevich, dan Donnelly (1996: 490) menyatakan informasi sebagai dasar untuk membuat keputusan yang efektif. Berdasarkan berbagai definisi yang telah diuraikan, dapat disimpulkan bahwa informasi adalah data yang diolah sehingga lebih berguna bagi penerima untuk mengambil suatu keputusan dan tindakan yang efektif.

The Institute of Chartered Accountants of India (2010) mengemukakan sistem secara istilah sebagai serangkaian unsur saling terkait yang tersusun secara teratur. Hartono (2005: 2) mendefinisikan sistem sebagai kumpulan dari unsur-unsur yang saling berinteraksi untuk mencapai suatu tujuan tertentu. Lucas dalam Kumorotomo dan Margono (1994: 8) mendefinisikan sistem sebagai suatu kumpulan unsur yang terorganisasi, saling berinteraksi, dan saling bergantung satu sama lain. Berdasarkan definisi yang telah diuraikan, dapat disimpulkan bahwa sistem merupakan serangkaian unsur yang tersusun secara

teratur, terorganisasi, saling terkait, saling berinteraksi, dan saling bergantung satu sama lain untuk mencapai suatu tujuan tertentu.

Gibson, Ivancevich, dan Donnelly (1996: 41) mendefinisikan teori sistem sebagai pendekatan analisis perilaku antar unsur sistem. *The Institute of Chartered Accountants of India* (2010) menggambarkan perilaku antar unsur sistem pada Gambar 1 yang terdiri dari *input* (pemasukan)*, processing* (pemrosesan)*, storage* (penyimpanan)*,* dan *output* (hasil). *Input* merupakan aliran data berasal dari luar sistem yang masuk ke suatu sistem untuk diproses atau diolah menjadi informasi, sehingga bermanfaat bagi pengguna sistem sesuai dengan tujuan. *Input* yang telah diolah tersebut disimpan dalam suatu penyimpanan dan sewaktu-waktu dapat ditampilkan dalam bentuk *output*.



Gambar 1. Hubungan antar unsur sistem

2. Sistem Manajemen Perpustakaan Sekolah

Undang-undang Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional dalam Suparman (2008: 54) mendefinisikan perpustakaan sebagai pusat sumber informasi ilmu pengetahuan. Perpustakaan adalah salah satu sarana penyediaan sumber-sumber informasi yang dapat digunakan untuk meningkatkan kualitas sumber daya manusia (Suharti, 2009: 1). Perpustakaan memiliki peran strategis dalam pengembangan ilmu pengetahuan dan peningkatan kualitas hidup manusia (Lasa, 2009: 69). Dalam Pasal 4 UU Nomor 43 Tahun 2007 tentang perpustakaan disebutkan bahwa perpustakaan bertujuan memberikan

layanan kepada pemustaka, meningkatkan kegemaran membaca, serta memperluas wawasan dan pengetahuan untuk mencerdaskan kehidupan bangsa.

Perpustakaan sekolah berfungsi sebagai sarana penunjang pendidikan di sekolah berupa penyediaan kumpulan bahan pustaka (Prastowo, 2013: 76). Dasar pembentukan perpustakaan sekolah adalah UU Sistem Pendidikan Nasional Nomor 2 Tahun 1989 dalam Dardiri, Septiyantono, dan Sidik (2001: 11), yang isinya menyatakan bahwa setiap sekolah harus menyediakan sumber belajar, yaitu perpustakaan. Perpustakaan sekolah diatur secara sistematis dalam satu ruang sehingga dapat membantu para siswa dan guru dalam proses pembelajaran (Prastowo, 2013: 76). Sejalan dengan tujuan perpustakaan secara umum, perpustakaan sekolah memiliki tujuan sebagai penyedia berbagai macam informasi kepada siswa dan guru untuk memperdalam pengalaman siswa dalam mempelajari ilmu pengetahuan dan membantu guru dalam melaksanakan kegiatan belajar mengajar di sekolah (Prastowo, 2013: 50).

Yujun, Hao, Tengfei, dan Zhiqiang (2012) menyatakan bahwa sebuah perpustakaan sekolah dikatakan profesional jika perpustakaan tersebut memberi kenyamanan bagi guru dan siswa dalam mengakses berbagai informasi yang ada di perpustakaan. Menurut Lasa (2009: 64), sumber-sumber informasi yang ada di perpustakaan dapat diakses secara optimal, mudah, dan cepat apabila perpustakaan memiliki manajemen yang profesional. Suparman (2008: 59) menyatakan bahwa tujuan perpustakaan dapat terealisasi dengan baik apabila didukung oleh sistem manajemen yang memadai.

Menurut Samsudin dalam Prastowo (2013: 19), kata manajemen berasal dari bahasa Inggris "*management*" yang dikembangkan dari kata "*to manage*"

yang berarti mengelola. Kumorotomo dan Margono (1994: 13) mendefinisikan manajemen sebagai kegiatan yang dilakukan di dalam suatu organisasi untuk mencapai tujuannya. Manajemen perpustakaan merupakan suatu usaha mengelola sumber daya perpustakaan (baik sumber daya manusia maupun sumber daya yang lain) melalui proses perencanaan untuk mencapai tujuan perpustakaan secara efisien dan efektif (Prastowo, 2013: 25). Efisien berarti penyelesaian suatu pekerjaan dilaksanakan secara tepat dan akurat tanpa membuang waktu, tenaga, dan biaya. Efektif berarti cara yang digunakan dalam mengelola perpustakaan adalah tepat sehingga tujuan yang diinginkan dapat dicapai secara maksimal (Prastowo, 2013: 23).

Berdasarkan berbagai definisi yang telah diuraikan, dapat disimpulkan bahwa sistem manajemen perpustakaan sekolah merupakan suatu usaha yang dilakukan oleh sekolah dalam mengelola berbagai sumber daya perpustakaan melalui proses perencanaan yang matang untuk memperluas wawasan dan pengetahuan para murid dan guru agar pelaksanaan kegiatan belajar mengajar dapat berjalan dengan efektif dan efisien.

3. Peran Pustakawan dalam Manajemen Perpustakaan

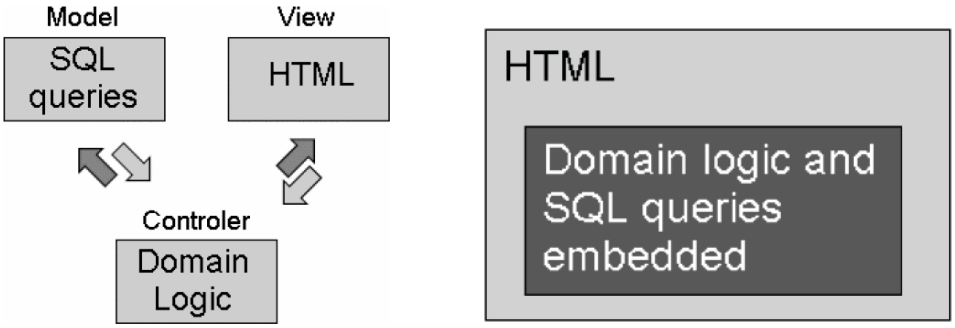
Pustakawan merupakan tenaga profesional yang berwewenang untuk mengelola perpustakaan (Suparman, 2008: 55). Dardiri, Septiyantono, dan Sidik (2001: 115) menyatakan kegiatan pelayanan oleh pustakawan sebagai salah satu unsur yang harus dipenuhi suatu perpustakaan. Pelayanan perpustakaan sekolah merupakan suatu upaya yang dilakukan oleh pustakawan sekolah agar bahan-bahan pustaka di perpustakaan sekolah dapat dimanfaatkan secara optimal oleh para penggunanya. Pelayanan suatu perpustakaan dikatakan prima (sangat baik)

jika para pengguna perpustakaan tersebut merasa puas dengan pelayanan yang berjalan di perpustakaan (Prastowo, 2013: 243-244). Pustakawan dapat melayani dengan baik jika puas dengan pekerjaannya. Pustakawan yang merasa puas dengan pekerjaannya akan memiliki sikap positif terhadap pekerjaan sehingga akan memotivasi dirinya untuk bekerja dengan optimal (Suharti, 2009: 4).

4. *Web Application Framework*

Web application framework merupakan perangkat lunak yang berisikan kerangka kerja untuk membangun perangkat lunak berbasis *Web*. Dewasa ini, mayoritas *Web application framework* yang beredar menggunakan teknik pemograman MVC atau *Model–View–Controller* (Vensada, 2011). Menurut Irena dan Bojan (2011), MVC memisahkan antara *controller* sebagai *domain logic*, *view* sebagai wadah untuk *interface*, dan *model* sebagai wadah untuk pemrosesan data. Menurut Wei, Lin, LiJing, dan Jing (2009), pemisahan komponen tersebut adalah pola yang efektif untuk membangun perangkat lunak berbasis *Web* sehingga bermanfaat bagi pengembang perangkat lunak untuk mengurangi kompleksitas ketika melakukan aktivitas pemograman (*coding*).

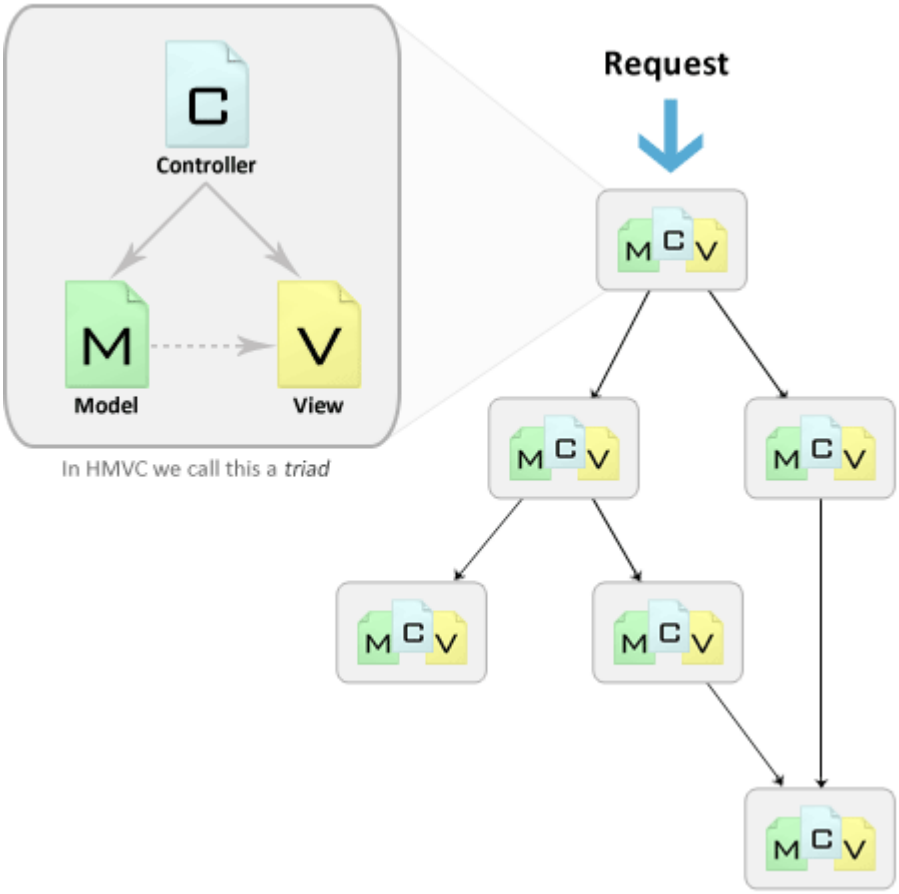
Irena dan Bojan (2011) menggambarkan perilaku antara *controller*, *model*, dan *view* pada Gambar 2. Seperti yang tampak pada Gambar 2, *model* dapat dikatakan sebagai *domain data*. *Model* biasanya digunakan untuk mengambil data dari basis data dan menyimpan data ke dalam basis data. *View* sebagai komponen yang berfungsi untuk menampilkan *interface*. *Controller* merupakan komponen yang digunakan untuk menangani interaksi dengan pengguna. *Controller* bekerja dengan *model* dan *view*. *Controller* merupakan penentu *view* yang akan digunakan (Sidik, 2012).



Gambar 2. MVC beserta hasil perilakunya

Davis dan Phillips (2007) mendefinisikan PHP atau *Hypertext Preprocessor* sebagai bahasa pemograman untuk membangun perangkat lunak berbasis *Web* secara interaktif. Hasil kerja *coding* dengan PHP disimpan pada suatu *Web server* agar dapat diakses. Salah satu dari *Web application framework* berbasis PHP dengan teknik pemograman MVC adalah *CodeIgniter* (CI). Menurut Sidik (2012), CI memiliki dokumentasi yang jelas dan lengkap, sehingga pengembang perangkat lunak dapat mempelajari dengan mudah. Jika dokumentasi yang tersedia dapat dipelajari dengan mudah, pengembang perangkat lunak diharapkan dapat membangun perangkat lunak dengan cepat.

Seiring dengan *Web application framework* yang terus berkembang, *Wiredesignz* menyediakan *extension* bagi pengguna CI untuk mengembangkan MVC menjadi HMVC (*Hierarchical Model–View–Controller*). HMVC merupakan solusi pengaturan MVC untuk mengurangi beban kerja (Cogan, 2010). Cai, Kapila, dan Pal (2000) mengemukakan bahwa HMVC memperjelas batasan antara *model, view,* dan *controller*. Berdasarkan Gambar 3 tentang pengaturan tata letak MVC pada HMVC, Cogan (2010) menyatakan "*a triad can request access to another triad via their controllers*" (melalui *controller*, suatu modul HMVC dapat mengakses komponen di modul HMVC yang lain).



Gambar 3. HMVC

5. *Software Engineering*

Kori, Ken-ichi, Kumiyo, Yoshihiro, Shingo, dan Kazayuki (1999) mendefinisikan *software engineering* sebagai aktivitas mengkaji kebutuhan calon pengguna untuk menghasilkan suatu perangkat lunak. *Software engineering* adalah aktivitas dalam rangka mengembangkan perangkat lunak dengan menggunakan berbagai metode, alat bantu, dan teknik tertentu (Bell, 2005). Agarwal, Tayal, dan Gupta (2010) mendefinisikan *software engineering* sebagai pendalaman semua aspek produksi perangkat lunak. *Software engineering* bertujuan agar produksi perangkat lunak dapat berjalan secara sistematis dan memuaskan pengguna. Shalahudin dan Rosa (2011: 4), mengemukakan tujuan *software*

engineering untuk menghasilkan perangkat lunak yang dapat bernilai dan bekerja secara efisien. Berdasarkan beberapa definisi yang telah diuraikan, dapat disimpulkan bahwa *software engineering* merupakan pengembangan perangkat lunak dengan berbagai metode, alat bantu, dan teknik tertentu untuk mendalami semua aspek produksi perangkat lunak dan mengkaji kebutuhan calon pengguna demi terciptanya suatu perangkat lunak yang dapat berjalan secara sistematis, bernilai, efisien, dan memuaskan calon pengguna. Pelaku *software engineering* disebut perekayasa (Sinamarta, 2010: 1).

Pressman (2010: 13-14) menggambarkan *software engineering* dalam bentuk lapisan yang terdiri dari *tools* (berbagai alat bantu), *methods* (metode-metode), *process* (proses), dan *quality focus* (kualitas). Produksi perangkat lunak tidak terbatas pada cara membangun perangkat lunak, tetapi juga mencakup berbagai pendekatan identifikasi masalah secara sistematis untuk diselesaikan dengan perangkat lunak (Sommerville, 2011). Lapisan *process* membahas berbagai pendekatan tersebut. Lapisan *process* berfungsi sebagai perekat terhadap lapisan *tools, methods,* dan *quality focus*. Lapisan *methods* menjelaskan secara teknis cara membangun perangkat lunak. Lapisan *methods* dan *process* didukung oleh lapisan *tools*. Pelaksanaan lapisan *tools, methods,* dan *process* secara berkesinambungan akan bermuara pada kualitas perangkat lunak.

Gambar 4. Lapisan pada *software engineering*

a. *Process*

Process merupakan fondasi dari *software engineering* (Pressman, 2010). Sommerville (2011) memberi istilah *process* pada *software engineering* sebagai *software process*. Osterweil dalam Sinamarta (2010: 38) menyatakan *process* meliputi berbagai aktivitas yang diperlukan untuk membangun perangkat lunak. *Process* memberikan suatu kerangka kerja yang berisikan rencana komprehensif untuk mengembangkan perangkat lunak (Al-Bahra, 2006: 34). Humprey dan Kellner dalam Sinamarta (2010: 38) menyatakan *process* sebagai kerangka kerja yang dibentuk untuk membangun perangkat lunak dengan menerapkan alat, metode, dan sumber daya manusia terkait. Berdasarkan berbagai definisi yang telah diuraikan, dapat disimpulkan bahwa *software process* merupakan sekumpulan kerangka kerja yang berisikan rencana komprehensif untuk membangun dan mengembangkan perangkat lunak dengan menerapkan alat, metode, dan sumber daya manusia terkait.

Pressman (2010) menetapkan lima kerangka kerja pada *software process*, yakni *communication* (pengkomunikasian)*, planning* (perencanaan)*, modeling* (pemodelan)*, construction* (pembangunan)*,* dan *deployment* (pendistribusian). Komunikasi kepada calon pengguna dilakukan sebagai identifikasi masalah. Komunikasi bermanfaat untuk mengetahui permasalahan yang terjadi dan harapan yang diinginkan oleh calon pengguna terhadap solusi yang akan ditawarkan oleh perekayasa pada kerangka kerja *planning*. Selanjutnya perekayasa melakukan pemodelan (*modeling*) untuk diimplementasikan pada kerangka kerja *construction* (pembangunan) dengan melakukan aktivitas pemograman (*coding*)

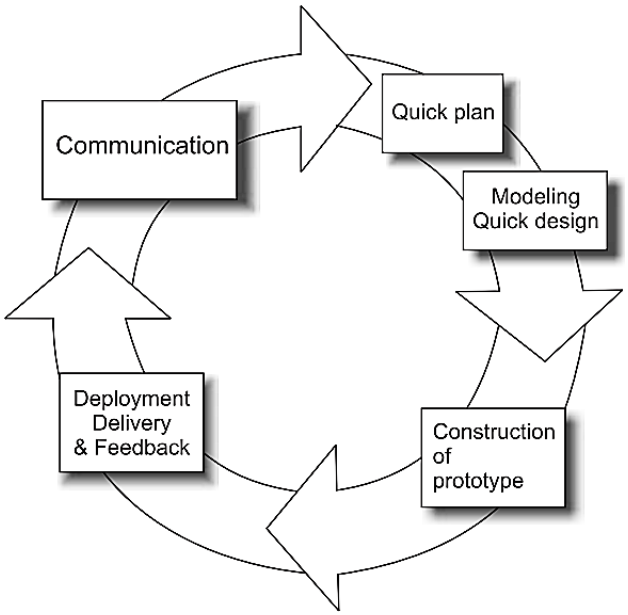
dan analisis kelayakan perangkat lunak. Perangkat lunak siap didistribusikan (*deployment*) kepada pengguna jika layak digunakan.

b. *Methods*

Agarwal, Tayal, dan Gupta (2010) mendefinisikan *methods* (metode-metode) sebagai panduan melaksanakan kegiatan *software engineering* secara tepat, sistematis, dan disiplin. *Methods* mencakup berbagai tugas, yaitu komunikasi, analisis kebutuhan, pemodelan untuk lapisan *process*, pembangunan perangkat lunak, pengujian perangkat lunak, dan dukungan perangkat lunak setelah selesai dibangun (Pressman, 2010). Bell (2005) memberi istilah pemodelan di atas sebagai *process model*. Menurut Bruegge dan Dutoit (2010), perekayasa harus tepat memilih satu model agar dapat fokus menghadapi berbagai kompleksitas pada *software engineering*.

Model *prototyping* merupakan salah satu dari jenis *process model* (Bell, 2005). Shalahuddin dan Rosa (2011: 29) mengemukakan bahwa model *prototyping* digunakan untuk mengatasi ketidakpahaman calon pengguna mengenai cara teknis membangun perangkat lunak dan memperjelas spesifikasi kebutuhan yang diinginkan calon pengguna kepada perekayasa. Jika dalam pelaksanaan model *prototyping* terjadi kesalahpahaman antara perekayasa dan calon pengguna, model *prototyping* memberikan solusi untuk melakukan identifikasi kesalahan sehingga dapat segera melakukan revisi (Al-Bahra, 2006: 25). Shalahuddin dan Rosa (2011: 31) mengemukakan bahwa model *prototyping* cocok digunakan untuk menjabarkan kebutuhan calon pengguna secara detil karena calon pengguna seringkali sulit menyampaikan kebutuhan secara detil.

Agarwal, Tayal, dan Gupta (2010) menjelaskan bahwa *prototyping* dimulai dari komunikasi antara perekayasa dan calon pengguna. Komunikasi digunakan sebagai identifikasi kebutuhan dan menyepakati tujuan pembuatan perangkat lunak. Langkah yang dilakukan setelah terjadi kesepakatan pada proses komunikasi adalah membuat rencana dan desain secara cepat untuk membangun perangkat lunak. Desain semaksimal mungkin dirancang sesuai dengan keinginan calon pengguna. Calon pengguna mengevaluasi perangkat lunak yang telah dibangun secara berulang hingga sesuai kebutuhan.

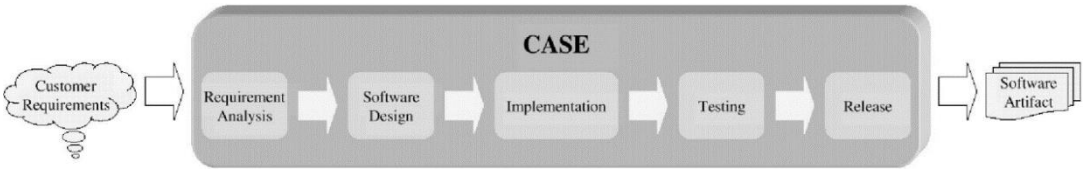


Gambar 5. Model *prototyping*

c. *Tools*

Menurut Pressman (2010), *tools* (berbagai alat bantu) pada *software engineering* disebut *Computer-aided Software Engineering* (CASE). Agarwal, Tayal, dan Gupta (2010) mengemukakan bahwa CASE berisikan berbagai alat bantu untuk mengembangkan dan memelihara perangkat lunak. Menurut Kori, Ken-ichi, Kumiyo, Yoshihiro, Shingo, dan Kazayuki (1999: 476-477), CASE ber-tujuan untuk memudahkan pengembang perangkat lunak dalam meningkatkan

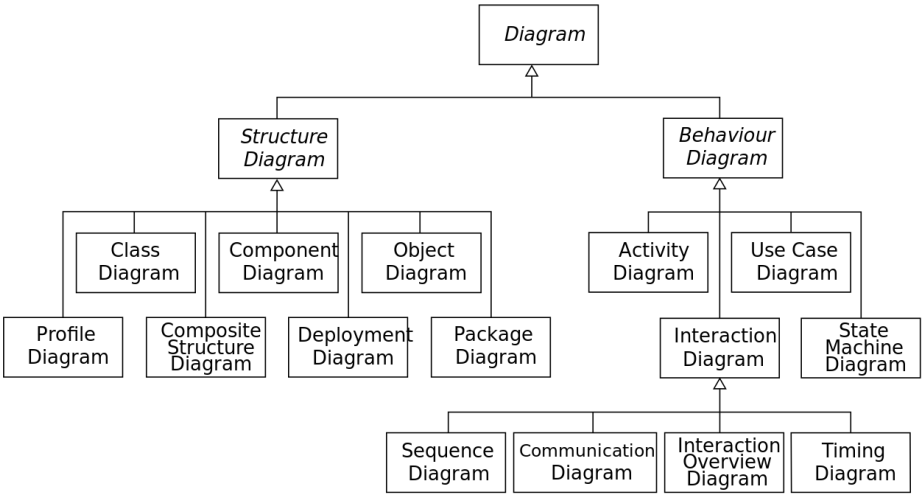
kecepatan dan kualitas produksi perangkat lunak. Peran CASE dalam *software engineering* digambarkan pada Gambar 6.



Gambar 6. Peran CASE dalam *software engineering*

Flowchart merupakan CASE pertama yang digunakan pada penelitian ini. Dickson dan Wetherbe (1985) mendefinisikan *flowchart* sebagai penggambaran representasi dari suatu sistem. *Flowchart* merupakan alat bantu untuk melakukan dokumentasi suatu sistem yang telah berjalan berdasarkan analisis yang dilakukan pada kerangka kerja *communication*. Berdasarkan hal tersebut, maka *flowchart* dapat digunakan untuk menggambarkan sistem manajemen yang telah berjalan di perpustakaan SD Muhammadiyah Condongcatur Yogyakarta. *Flowchart* digunakan sebagai acuan untuk merancang *Unified Modeling Language* (UML), *interface*, dan basis data.

UML (*Unified Modeling Language*) merupakan CASE kedua yang digunakan pada penelitian ini. UML merupakan bahasa standar pemodelan perangkat lunak (Pressman, 2010). Shalahudin dan Rosa (2011: 118) mendefinisikan UML sebagai bahasa visual untuk pemodelan dan komunikasi sistem berupa diagram beserta teks-teks pendukung. Dari berbagai definisi yang telah diuraikan dapat disimpulkan bahwa UML merupakan bahasa standar untuk pemodelan dan komunikasi sistem suatu perangkat lunak berupa diagram beserta teks-teks pendukung. *Object Management Group* atau OMG (2011) menjabarkan pembagian kategori UML beserta jenisnya pada Gambar 7.



Gambar 7. Kategori UML beserta jenisnya

Shalahudin dan Rosa (2011: 121) mendefinisikan *structure diagram* sebagai diagram yang menjelaskan struktur sistem. *Structure diagram* yang digunakan dalam penelitian ini adalah *component diagram*. Chumpol dan Wiwat (2013) mengemukakan *component diagram* sebagai salah satu bahasa populer untuk menggambarkan model arsitektural perangkat lunak. Menurut Schach (2008), *component diagram* digunakan untuk menggambarkan ketergantungan antar komponen pada suatu perangkat lunak.

Shalahudin dan Rosa (2011: 121) mendefinisikan *behavior diagram* sebagai diagram yang menjelaskan perilaku sistem. *Behavior diagram* yang digunakan dalam penelitian ini adalah *use case diagram*. *Use case diagram* terdiri dari dua komponen kunci, yaitu aktor (objek) dan *use case* (Bui, 2007). Malan dan Bredemeyer dalam Gretchen dan Daniel (2005) mengemukakan bahwa *use case diagram* menggambarkan perilaku objek dalam suatu sistem untuk mencapai tujuan tertentu. Menurut Miles dan Hamilton (2006), *use case diagram* saja tidak cukup untuk menggambarkan desain sistem suatu perangkat lunak yang akan dibuat. Maka dari itu, *use case diagram* perlu dijelaskan dengan *use*

case description yang berisikan informasi penting berupa deskripsi teks tentang *use case diagram*. Stéphane (2007) mencontohkan penjelasan yang harus ada dalam *use case desciption* pada Gambar 8.

Title: SupplierA Bid
System Under Design: Broker System
Precondition: An Order has been broadcasted
Steps
1.SupplierA receives the order and examines it
2.SupplierA submits a bid for order
3.The System shows the Bid to the Customer
Alternatives
1.a.SupplierA can not satisfy the Order
1.a.1.SupplierA passes on the Order
Success Postcondition: SupplierA has submitted a bid

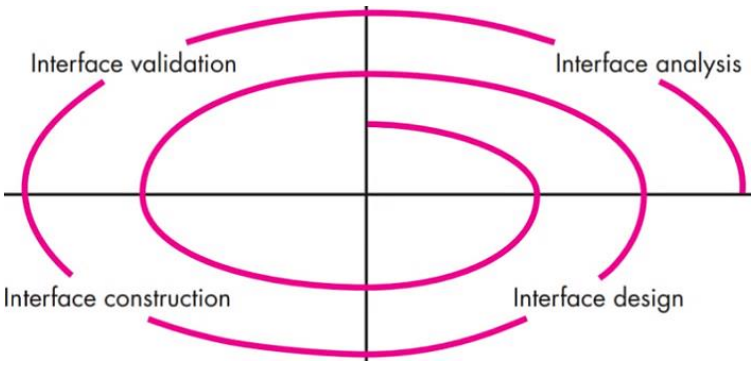
Gambar 8. *Use case description*

Shalahudin dan Rosa (2011: 121) mendefinisikan *interaction diagram* sebagai diagram yang menggambarkan interaksi sistem. Menurut R. Rönnquist dan C.K. Low (1996), *interaction diagram* digunakan untuk menggambarkan komunikasi dan aktivitas komputasi antar objek (aktor). *Interaction diagram* merupakan bagian dari *behavior diagram*. Artinya, *interaction diagram* tidak dapat dirancang jika tidak didahului dengan merancang salah satu jenis dari *behavior diagram*.

Interaction diagram yang akan digunakan dalam penelitian ini adalah *sequence diagram*. Menurut Shalahudin dan Rosa (2011: 137), *sequence diagram* menggambarkan perilaku objek (aktor) terhadap *use case*. Oleh karena itu, hal yang harus dilakukan sebelum merancang *sequence diagram* adalah mengetahui objek yang terlibat pada *use case diagram*.

Booch *et. al.* (2007) mengemukakan bahwa dalam rancangan *sequence diagram*, objek-objek yang terlibat pada suatu sistem digambarkan secara horizontal. Perilaku antar objek digambarkan dengan garis dan notasi tertentu untuk menunjukkan perilaku. Sebuah garis putus-putus digambarkan di bawah setiap objek untuk menjembatani penggambaran perilaku antar objek.

Perancangan *interface* merupakan CASE ketiga yang digunakan pada penelitian ini. Menurut Pressman (2010), perancangan *interface* membantu perekayasa untuk menciptakan interaksi yang efektif antara calon pengguna dan perangkat lunak Seperti yang tampak pada Gambar 9, perancangan *interface* terdiri dari empat kerangka kerja. Sebagai kerangka kerja pertama, *interface analysis* dilakukan dengan menganalisis latar belakang calon pengguna. Latar belakang pengguna tersebut dijadikan acuan untuk melakukan *task analysis* yang akan diselesaikan oleh calon pengguna. Hasil dari *task analysis* tersebut akan dimplementasikan pada kerangka kerja *interface design* dengan menentukan berbagai aksi dan objek yang akan digunakan calon pengguna dalam menyelesaikan tugasnya. *Interface design* harus memenuhi unsur *usability* (mudah dipakai). Hasil kerja dari *interface design* tersebut dimplementasikan pada kerangka kerja *interface construction* sebagai bagian dari *coding*. Calon pengguna melakukan evaluasi *interface* secara berulang hingga mencapai *interface* yang diinginkan. Setelah selesai dibangun, *interface* akan dievaluasi pada kerangka kerja *interface validation* hingga memenuhi dua hal, yaitu: (a) Mampu memenuhi semua tugas yang dibutuhkan oleh pengguna (b) Pengguna mudah dalam memahami dan menggunakan perangkat lunak.



Gambar 9. Kerangka kerja perancangan *interface*

Perancangan basis data merupakan CASE terakhir yang digunakan pada penelitian ini. Basis dapat diartikan sebagai tempat berkumpul. Basis data merupakan kumpulan data yang saling berhubungan dan disimpan untuk memenuhi berbagai kebutuhan. Kumpulan data tersebut diorganisasi agar setiap saat dapat dimanfaatkan dengan cepat dan mudah. (Fathansyah, 2012: 2-3). Perancangan basis data diperlukan untuk menciptakan basis data yang efisien dalam penggunaan ruang penyimpanan, cepat dalam pengaksesan, dan mudah dalam pemanipulasian (Fathansyah, 2012: 41). Salah satu model yang dapat digunakan untuk merancang basis data adalah *crow's foot* (Powell, 2006).

d. *Quality*

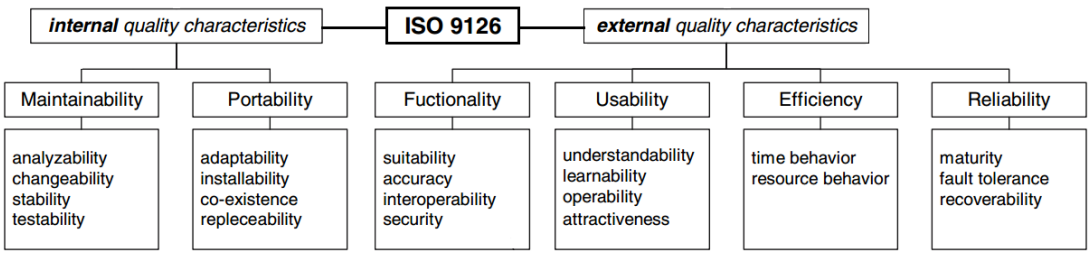
Agarwal, Tayal, dan Gupta (2010) menyatakan bahwa tujuan dasar dari kegiatan *software engineering* adalah menghasilkan perangkat lunak yang berkualitas. ISO/IEC (*International Organization for Standardization and The International Electrotechnical Commission*) dalam Antonia dan Michalis (2008) mendefinisikan kualitas sebagai kemampuan sekumpulan fitur dan karakteristik pada suatu produk dalam memenuhi kebutuhan penggunanya. Sinamarta (2010) menggambarkan peran karakteristik dan kebutuhan sebagai penentu kualitas suatu produk pada Gambar 10.

Gambar 10. Peran karakteristik dan kebutuhan sebagai penentu kualitas

6. Evaluasi Kualitas *Web* Berdasarkan ISO/IEC 9126

Suchman dalam Arikunto dan Jabar (2010: 1) mendefinisikan evaluasi sebagai sebuah proses menentukan hasil dari beberapa kegiatan yang direncanakan untuk mendukung tercapainya tujuan. Arikunto dan Jabar (2010: 17) mengatakan bahwa dengan evaluasi, suatu program dapat dicari komponen yang tidak bekerja dengan semestinya. Arikunto dan Jabar (2010: 29) mengatakan bahwa hasil evaluasi menentukan lanjut atau tidaknya dari suatu program.

Jae, Jung, Du, dan Soo (2009) menyatakan ISO/IEC 9126 sebagai standar internasional untuk mengevaluasi kualitas suatu produk. Mbusi dan Cornelis (2007) menyatakan bahwa ISO/IEC 9126 dapat digunakan untuk mengevaluasi kualitas berbagai macam jenis perangkat lunak. Hal tersebut menjadi dasar bagi Mahmoud dan Salah (2011) menggunakan ISO/IEC 9126 untuk mengevaluasi kualitas perangkat lunak berbasis *Web*.



Gambar 11. Standar kualitas ISO/IEC 9126

Pembagian kategori ISO/IEC 9126 beserta karakteristik dan sub-karakteristiknya digambarkan oleh Antonia dan Michalis (2008) pada Gambar 11. Berdasarkan Gambar 11, ISO/IEC 9126 terdiri dari dua aspek, yaitu internal dan eksternal. Kualitas internal terdiri dari *maintainability* dan *portability*. Kualitas eksternal terdiri dari *functionality, usability, efficiency, dan reliability*. Menurut Antonia dan Michalis (2008), kualitas internal merupakan kualitas dari sudut

pandang pengembang perangkat lunak dalam rangka memenuhi kebutuhan pengguna. Kualitas eksternal merupakan kualitas dari sudut pandang pengguna dalam menggunakan perangkat lunak.

a. *Maintainability*

Menurut ISO/IEC (2000), *maintainability* merupakan kemampuan perangkat lunak untuk dimodifikasi yang mencakup perbaikan dan adaptasi perangkat lunak. Sub-karakteristik *maintainability* adalah sebagai berikut:

1) *Analysability*: Kemampuan perangkat lunak untuk dianalisa dengan mudah jika terjadi kesalahan pengoperasian.
2) *Changeability*: Kemampuan perangkat lunak untuk dimodifikasi.
3) *Stability*: Kemampuan perangkat lunak untuk menghindari hal tidak terduga akibat dari modifikasi.
4) *Testability*: Kemampuan perangkat lunak untuk melakukan validasi akibat dari modifikasi.

Ilja, Tobias, dan Joost (2007) mengkritisi *metrics* pada ISO/IEC 9126 *maintainability* sebagai *metrics* dengan *lack predictive power* (kemampuan prediksi sangat lemah). *Metrics* tersebut tidak berdasarkan sudut pandang subjek *maintenance* perangkat lunak, tetapi berdasarkan sudut pandang teknisi perangkat lunak melakukan *maintenance*. Oleh karena itu, Ilja, Tobias, dan Joost (2007) membangun metode alternatif berupa pemetaan sub-karakteristik pada ISO/IEC 9126 *maintainability* dengan properti salah satu subjek *maintenance* perangkat lunak, yaitu *source code properties*. Tanda silang pada Gambar 12 menunjukkan bahwa kualitas properti *source code* berpengaruh besar terhadap kualitas sub-karakteristik pada ISO/IEC 9126 *maintainability*.

ISO 9126 maintainability	source code properties				
	volume	complexity per unit	duplication	unit size	unit testing
analysability	x		x	x	x
changeability		x	x		
stability					x
testability		x		x	x

Gambar 12. Pemetaan *maintainability* ke *source code properties*

Berdasarkan pemetaan tersebut, dapat disimpulkan bahwa evaluasi berbagai sub-karakteristik pada ISO/IEC 9126 *maintainability* dilakukan dengan cara mengevaluasi kualitas properti *source code* yang terangkum dalam suatu perhitungan bernama *Maintainability Index* (MI). Don, Dan, Bruce, dan Paul (1994) menjabarkan rumus MI sebagai berikut:

$$171 - 5.2 * \ln(\text{aveV}) - 0.23 * \text{aveV}(g') - 16.2 * \ln(\text{aveLOC}) + 50 \sin\sqrt{2.46 * \text{perCM}}$$

aveV = Rata-rata *Halstead Volume* per modul *source code*

aveV(g') = Rata-rata *Cyclomatic Complexity* per modul *source code*

aveLOC = Rata-rata *Line of Codes* per modul *source code*

perCM = Persentase komentar per modul *source code*

Welker dan Oman dalam Aldo (2001) menyarankan untuk tidak memasukkan komentar ke rumus MI karena dalam beberapa situasi, komentar menjadi tidak sinkron terhadap *source code* sehingga menurunkan hasil MI. Berdasarkan saran tersebut, rumus MI yang paling tepat adalah sebagai berikut:

$$171 - 5.2 * \ln(\text{aveV}) - 0.23 * \text{aveV}(g') - 16.2 * \ln(\text{aveLOC})$$

Menurut Muhammad, Wasif, Imran, dan Waqar (2008), perhitungan MI dapat dilakukan dengan bantuan *MI tools*. Salah satunya adalah *The Source*

Code Engine dari *Semantic Designs*. Hasil perhitungan berupa *Halstead Volume*, *Cyclomatic Complexity*, dan LOC oleh *tool* tersebut, dimasukkan ke rumus MI untuk mengevaluasi kualitas perangkat lunak dari sisi *maintainability*.

b. *Portability*

Menurut ISO/IEC (2000), *portability* merupakan kemampuan perangkat lunak untuk dipindah dari suatu tempat ke tempat lain. Sub-karakteristik *portability* adalah sebagai berikut:

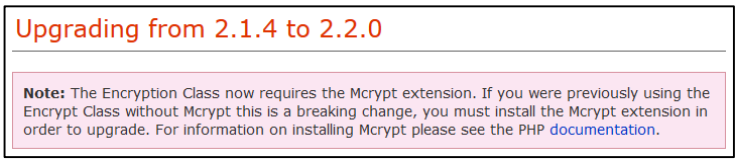
1) *Adaptability*: Kemampuan perangkat lunak beradaptasi dengan lingkungan tertentu yang berbeda dari lingkungan sebelumnya.
2) *Installability*: Kemampuan perangkat lunak dipasang di lingkungan tertentu.
3) *Co-existence*: Kemampuan perangkat lunak dapat aktif berdampingan dengan perangkat lunak lainnya dalam suatu lingkungan yang sama.
4) *Replaceability*: Kemampuan suatu perangkat lunak untuk dapat diperbaharui.

Nations (2014) mengemukakan *Web* sebagai perangkat lunak yang dapat diakses oleh *client* berupa *Web browser*. Artinya, perangkat lunak berbasis *Web* hanya dapat diakses dengan *Web browser*. Jika dikorelasikan dengan definisi masing-masing sub-karakteristik pada ISO/IEC 9126 *portability*, maka dapat terbentuk definisi sebagai berikut:

1) *Adaptability*: Kemampuan *Web* beradaptasi dengan *Web browser* yang berbeda dari *Web browser* sebelumnya.
2) *Installability*: Kemampuan *Web* dipasang di *Web browser* tertentu.
3) *Co-existence*: Kemampuan *Web* dapat aktif berdampingan dengan aplikasi lainnya dalam suatu lingkungan yang sama.
4) *Replaceability*: Kemampuan *Web* untuk dapat diperbaharui.

Jika diperhatikan dengan cermat, definisi *co-existence* sendiri dapat relevan dengan definisi pada salah satu sub-karakteristik pada *functionality*, yaitu *interoperability*. *Interoperability* merupakan kemampuan *Web* dapat berinteraksi dengan berbagai aplikasi yang dapat berupa basis data (Pressman, 2010). Pada penelitian ini, *Web* dan basis data aktif dan berada di suatu lingkungan yang sama, dalam hal ini adalah *Web server*. Atas dasar tersebut, dapat disimpulkan bahwa *co-existence* pada penelitian ini diwakilkan pada *interoperability*.

Penelitian ini membangun perangkat lunak berbasis *Web* dengan CI atau *CodeIgniter* yang dikeluarkan oleh *EllisLab, Inc*. Pembaharuan (*upgrading*) CI dapat dilakukan jika *EllisLab, Inc*. mengeluarkan rilis CI baru. Pada setiap rilis yang baru, *EllisLab, Inc*. memberikan panduan untuk melakukan pembaharuan CI. Berdasarkan hal tersebut, maka dapat disimpulkan bahwa evaluasi dari sisi *replaceability* tidak perlu dilakukan.


Upgrading from 2.1.4 to 2.2.0

Note: The Encryption Class now requires the Mcrypt extension. If you were previously using the Encrypt Class without Mcrypt this is a breaking change, you must install the Mcrypt extension in order to upgrade. For information on installing Mcrypt please see the PHP documentation.

Gambar 13. Panduan pembaharuan *CodeIgniter*

Berdasarkan kajian teori yang telah diuraikan, maka sub-karakteristik pada *portability* yang dapat dievaluasi adalah *adaptability* dan *installability*. Jika diperhatikan dengan cermat, sesungguhnya definisi *adaptability* dan *installability* bermuara pada definisi *portability*, yaitu kemampuan *Web* untuk dipindah dari suatu *Web browser* ke *Web browser* yang lain. Berdasarkan hal tersebut, dapat disimpulkan bahwa evaluasi *portability* pada *Web* cukup menggunakan cara oleh Ville (2012), yaitu membandingkan penggunaan perangkat lunak *Web* dengan berbagai jenis *Web browser*.

c. *Functionality*

Menurut ISO/IEC (2000), *functionality* merupakan kemampuan perangkat lunak menyediakan fungsi yang dibutuhkan ketika perangkat lunak dijalankan. Sub-karakteristik *functionality* adalah sebagai berikut:

1) *Suitability*: Kemampuan perangkat lunak menyediakan fungsi untuk memenuhi berbagai tugas dan tujuan pengguna.
2) *Accuracy*: Kemampuan perangkat lunak menyediakan hasil yang akurat dan sesuai dengan kebutuhan.
3) *Interoperability*: Kemampuan antar sistem dan atau satu sistem dengan sistem lainnya pada suatu perangkat lunak untuk saling berinteraksi.
4) *Security*: Kemampuan perlindungan data suatu perangkat lunak dari akses tidak sah, sehingga orang yang tidak berwenang tidak dapat merubah atau memodifikasi data pada suatu perangkat lunak.

Menurut Pressman (2010), evaluasi kualitas *functionality* suatu *Web* berfungsi untuk mengetahui tingkat kesesuaian hasil suatu *Web* terhadap kebutuhan calon pengguna. Sub-karakteristik yang relevan dengan pernyataan tersebut adalah *interoperability* dan *suitability*. *Interoperability* dievaluasi untuk memastikan antar komponen perangkat lunak dapat berinteraksi dengan benar (sesuai rencana). Jika *interoperability* baik, maka *Web* dapat menampilkan hasil yang akurat (*good accuracy*). *Suitability* dievaluasi untuk memastikan bahwa fungsi yang telah disediakan sesuai dengan kebutuhan pengguna.

Menurut Andrea (2005), evaluasi *interoperability* suatu *Web* dapat dilakukan dengan *test cases* berdasarkan UML (*Unified Modeling Language*) yang berisikan *pre-conditions* dan *post-conditions*. UML yang berisikan *pre-conditions*

dan *post-conditions* adalah *use case descriptions* (Stéphane, 2007). UML akan divalidasi konstruksinya terhadap perangkat lunak yang telah dibangun. Validasi konstruksi menggunakan pendapat ahli dengan jumlah minimal tiga orang. Ahli berupa orang yang menguasai lingkup penelitian (Sugiyono, 2013: 125). Atas dasar tersebut, ahli yang relevan dengan penelitian ini adalah ahli *software engineering* dan ahli perangkat lunak berbasis *Web*.

Evaluasi *suitability* pada penelitian ini menggunakan *Suitability Evaluation Questionnaire* (SEQ) berskala *Likert* lima level oleh José-Antonio Gil-Gómez, Herme, José-Antonio Lozano-Quilis, Pilar, Sergio, dan Carmen (2013) dengan sedikit penyesuaian untuk Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta. Responden evaluasi *suitability* berupa pustakawan. Tingkat persetujuan responden pada *suitability* dapat diketahui dengan teknik perhitungan kontinum oleh Sugiyono (2013).

Selain *interoperability* dan *suitability*, sub-karakteristik yang penting untuk dievaluasi adalah *security* (keamanan). Menurut Acunetix (2014), *security* suatu *Web* pada dewasa ini menjadi perhatian yang sangat serius. Jika celah keamanan suatu *Web* berhasil ditembus, maka konten dari *Web* tersebut dapat dirubah sesuka hati oleh peretas. Hal tersebut mengakibatkan tampilan *Web* yang tidak sesuai dengan keinginan pengguna. Evaluasi celah keamanan suatu *Web* dapat dilakukan dengan bantuan *tool*. Menurut Marco, Nuno, dan Henrique (2009), salah satu *tool* yang dapat mendeteksi celah keamanan suatu *Web* adalah *Acunetix Web Vulnerability Scanner*. Sebagai aplikasi berbayar, *Acunetix Web Vulterability Scanner* dapat mengevaluasi keamanan *Web* lebih tajam daripada aplikasi serupa bersifat gratis seperti *WSDigger* dan *wsfuzzer*.

d. *Usability*

Menurut ISO/IEC (2000), *usability* adalah kemampuan perangkat lunak untuk dipahami dengan mudah, dipelajari dengan mudah, digunakan dengan mudah, dan menarik bagi pengguna dalam berbagai kondisi. Sub-karakteristik *usability* adalah sebagai berikut:

1) *Understandability*: Kemampuan perangkat lunak untuk dipahami dengan mudah sehingga pengguna terbantu dalam menyelesaikan tugasnya.
2) *Learnability*: Kemampuan perangkat lunak untuk dipelajari.
3) *Operability*: Kemampuan perangkat lunak untuk dijalankan.
4) *Attractiveness*: Kemampuan perangkat lunak untuk dapat dipandang menarik bagi pengguna.

Penelitian ini menggunakan *USE questionnaire* oleh Lund (2001) dengan skala *Likert* tujuh level. *USE questionnaire* terdiri dari empat sub-karakteristik, yaitu *usefulness*, *ease of use*, *ease of learning*, dan *satisfaction*. *USE questionnaire* relevan dengan ISO/IEC 9126 *usability* karena *usefulness* relevan dengan ISO/IEC 9126 *operability*, *ease of use* relevan dengan ISO/IEC 9126 *understandability*, *ease of learning* relevan dengan ISO/IEC 9126 *learnability*, dan *satisfaction* relevan dengan ISO/IEC 9126 *attractiveness*.

Jakob dan Thomas (1993) memaparkan hasil penelitian perolehan standar deviasi berdasarkan jumlah responden dalam mengevaluasi *usability* pada Tabel 1. Halina, Ismail, dan Abdul (2010: 67) menyatakan "*The low standard deviations means the most observations center around the mean*" (standar deviasi rendah berarti jawaban responden tidak jauh berbeda dengan rata-rata). Berdasarkan pernyataan tersebut dapat disimpulkan bahwa penelitian kuantitatif semakin

akurat jika standar deviasi semakin kecil. Berdasarkan Tabel 1, evaluasi *usability* dapat dikatakan akurat jika jumlah responden minimal sembilan orang.

Tabel 1. Perolehan standar deviasi berdasarkan jumlah responden *usability*

Jumlah responden	Standar deviasi
3	44%
4	21%
5	11%
6	9%
7	8%
8	6%
9	5%
10	5%

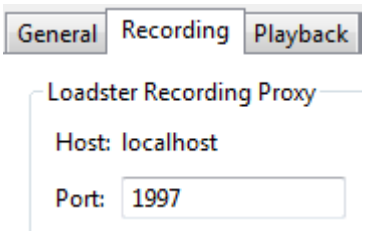
e. *Efficiency*

Menurut ISO/IEC (2000), *efficiency* merupakan kemampuan perangkat lunak dapat bekerja dengan relatif baik sesuai dengan *resources* yang telah ditetapkan. Sub-karakteristik *efficiency* adalah sebagai berikut:

1) *Time behavior*: Nominal waktu suatu perangkat lunak dalam memberikan respon dari permintaan pengguna.
2) *Resource utilisation*: Kemampuan perangkat lunak menggunakan *resources* berdasarkan ketentuan yang telah ditetapkan.

ISO/IEC (2002) menetapkan *response time metric* untuk *time behavior* dengan ketentuan bahwa suatu produk dikatakan baik dari sisi *time behavior* jika nilai T (nominal waktu) mendekati 0, dengan 0 < T. ISO/IEC (2002) menetapkan *user waiting time of* I/O (*Input/Output*) *devices utilisation metric* untuk evaluasi *resource utilisation* dengan ketentuan bahwa suatu produk dikatakan semakin baik dari sisi *resource utilisation* jika nilai T mendekati 0, dengan 0 < T. Berdasarkan kesamaan kriteria T antara sub-karakteristik *time behavior* dan *resource utilisation*, maka evaluasi *efficiency* dapat diwakilkan oleh salah satu *metric*, yaitu *response time metric* pada sub-karakteristik *time behavior*.

Pooja dan Sanjay (2013) menyatakan *Loadster* sebagai *tool* terbaik untuk mengevaluasi kinerja *Web*. Salah satu evaluasi yang dapat dilakukan oleh *Loadster* adalah evaluasi *response time*. Kelebihan *Loadster* terletak pada metode rekam yang dapat diterima oleh semua *Web browser* berbasis *desktop*. Metode tersebut adalah merekam berdasarkan pengaturan *proxy* pada *Web browser*. Berdasarkan metode tersebut, pengguna cukup menyamakan pengaturan *proxy* antara *Web browser* dan *tool Loadster*.



Gambar 14. Pengaturan *proxy* pada *tool Loadster*

f. *Reliability*

Menurut ISO/IEC (2002), *reliability* merupakan kemampuan perangkat lunak mempertahankan kinerjanya ketika dijalankan. Sub-karakteristik *reliability* adalah sebagai berikut:

1) *Maturity*: Kemampuan perangkat lunak mengantisipasi kegagalan.
2) *Fault tolerance*: Kemampuan perangkat lunak bertahan jika terjadi kesalahan dalam pengoperasian.
3) *Recoverability*: Kemampuan perangkat lunak untuk bangkit kembali dan memulihkan data akibat dari kesalahan dalam pengoperasian.

ISO/IEC (2002) menetapkan *test maturity metric* untuk evaluasi *maturity* dengan rumus sebagai berikut:

$$X = \frac{A}{B} = \frac{\text{Jumlah } test\ cases \text{ gagal}}{\text{Jumlah } test\ cases \text{ yang dievaluasi}}$$

Rumus untuk evaluasi *maturity* tersebut akan menghasilkan nilai 0 ≤ X. Suatu produk dikatakan semakin baik dari sisi *maturity* jika nilai X mendekati 0.

ISO/IEC (2002) menetapkan *failure avoidance metric* untuk evaluasi *fault tolerance* dengan rumus sebagai berikut:

$$\mathrm{X} = \frac{\mathrm{A}}{\mathrm{B}} = \frac{\text{Jumlah } test\ cases \text{ gagal yang berhasil dihindari}}{\text{Jumlah } test\ cases \text{ yang dievaluasi}}$$

Rumus untuk evaluasi *fault tolerance* tersebut akan menghasilkan nilai 0 ≤ X ≤ 1. Suatu produk dikatakan semakin baik dari sisi *fault tolerance* jika nilai Y mendekati 1.

ISO/IEC (2002) menetapkan *availability metric* untuk evaluasi *recoverability* dengan rumus sebagai berikut:

$$\mathrm{Y} = \frac{\mathrm{A1}}{\mathrm{A2}} = \frac{\text{Jumlah } test\ cases \text{ sukses}}{\text{Jumlah } test\ cases \text{ yang dievaluasi}}$$

Rumus untuk evaluasi *recoverability* tersebut akan menghasilkan nilai 0 ≤ Y ≤ 1. Suatu produk dikatakan semakin baik dari sisi *recoverability* jika nilai Y mendekati 1.

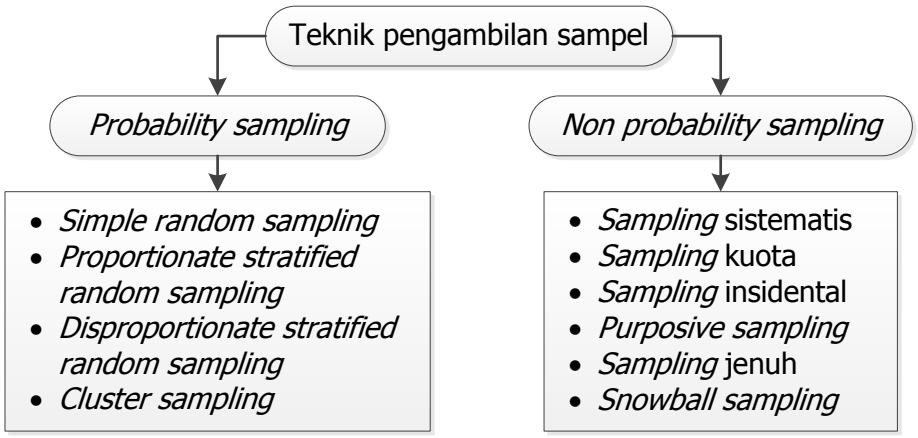
Jika diperhatikan secara seksama, antar rumus dan kriteria nilai tersebut saling berkaitan. Jika salah satu sub-karakteristik pada *reliability* hasilnya sangat baik, maka sub-karakteristik pada *reliability* yang lainnya secara otomatis juga sangat baik. Sebagai contoh, jumlah *test cases* yang dievaluasi sebanyak 73 dengan keberhasilan 100%. Jika dimasukkan ke rumus *maturity*, maka X = 0 (sangat baik). Jika dimasukkan ke rumus *fault tolerance*, maka X = 1 (sangat baik). Jika dimasukkan ke rumus *recoverability*, maka Y = 1 (sangat baik). Untuk hasil lebih akurat, *Telcordia* dalam Abhaya dan Jack (2009) menyatakan bahwa 95% dari semua *test cases* pada evaluasi *reliability* harus lolos.

Atas dasar keterkaitan antar *metric* tersebut, evaluasi kualitas dari sisi *reliability* berdasarkan sub-karakteristiknya dapat diwakilkan salah satu sub-karakteristik pada *reliability*. Menurut Luis, Guillermo, dan Gustavo (2006), sub-karakteristik *maturity* dapat digunakan untuk mengevaluasi *Web* dari sisi *reliability*. ISO/IEC (2002) merekomendasikan evaluasi *maturity* dilaksanakan dengan metode *stress testing*. Pooja dan Sanjay (2013) menyatakan *Loadster* sebagai salah satu *stress testing tools* untuk *Web*. Hasil dari *Loadster* tersebut akan menjadi bahan evaluasi kualitas *Web* dari sisi *maturity*.

7. Teknik Pengambilan Sampel

Salah satu aspek yang sangat penting dalam suatu penelitian adalah teknik pengambilan sampel. Teknik pengambilan sampel dibagi menjadi dua golongan, yaitu *probability sampling* dan *nonprobability sampling*. *Probability sampling* adalah teknik pengambilan sampel yang memberikan peluang sama bagi anggota populasi untuk dipilih menjadi sampel. *Nonprobability sampling* adalah teknik pengambilan sampel yang tidak memberi peluang sama bagi setiap unsur populasi untuk dipilih menjadi sampel (Sugiyono, 2013: 82-84).

Sugiyono (2013) memaparkan teknik-teknik pengambilan sampel pada Gambar 15. Penelitian ini menggunakan *purposive sampling* sebagai teknik pengambilan sampel. *Purposive sampling* adalah teknik penentuan sampel dengan pertimbangan tertentu. Sebagai contoh, seorang peneliti akan melakukan penelitian tentang kualitas makanan, maka sampelnya adalah orang yang ahli makanan (Sugiyono, 2013: 85). Atas dasar tersebut, maka sampel yang tepat untuk penelitian ini adalah ahli *software engineering*, ahli perangkat lunak berbasis *Web*, dan ahli manajemen perpustakaan.



Gambar 15. Teknik-teknik pengambilan sampel

B. Kajian Penelitian yang Relevan

1. David (2006) membuat sistem untuk membantu Universitas Surakarta dalam menyelesaikan masalah pengelolaan data buku dan anggota. Kelemahan dari penelitian ini adalah sistem tidak dapat membuat nomor buku dan nomor anggota yang bersifat *unique* secara otomatis berdasarkan berbagai atribut pada buku dan anggota tersebut. Pustakawan harus membuat nomor anggota dan nomor buku secara manual. Hal ini tidak ada bedanya dengan membuat nomor anggota dan nomor buku bersifat *unique* secara tertulis. Permasalahan tersebut dapat diatasi dengan membuat sistem yang dapat membuat nomor buku dan nomor anggota secara otomatis berdasarkan atribut yang telah dimasukkan. Hal ini diharapkan dapat memudahkan pustakawan dalam melakukan pengkategorian data anggota dan data buku.
2. Derta (2013) membuat sistem untuk membantu SMA N 3 Pemalang dalam menyelesaikan permasalahan sulitnya mendeteksi buku yang belum dikembalikan karena belum ada sistem yang efektif dan efisien dalam mengelola perpustakaan. Dalam penelitiannya, Derta berhasil menyelesaikan masalah deteksi buku tersebut dengan menyediakan pencarian data buku. Kelemahan

dari sistem ini adalah pencarian buku yang hanya terdiri dari satu atribut saja, yaitu judul buku. Kelemahan ini juga terjadi pada pencarian data anggota yang hanya disediakan pencarian berdasarkan nama anggota. Permasalahan tersebut dapat diatasi dengan menyediakan pencarian berbagai atribut. Sebagai contoh, pada data buku, peneliti akan menyediakan pencarian untuk atribut nomor buku, judul buku, penulis, penerbit, dan lain-lain.

3. Ervira (2014) membuat sistem untuk membantu Perpustakaan Dakwah Kampus UGM dalam menyelesaikan permasalahan sirkulasi. Kelemahan dari penelitian ini adalah sistem yang dibuat Ervira belum efektif dalam menyelesaikan permasalahan sirkulasi, salah satunya pada waktu peminjaman buku. Pustakawan harus memasukkan tanggal pinjam pada sistem sebagai perhitungan dimulainya anggota perpustakaan dalam meminjam buku. Hal ini tidak ada bedanya dengan mencatat tanggal peminjaman secara manual. Permasalahan tersebut dapat diatasi dengan membuat sistem yang dapat mendeteksi tanggal peminjaman secara otomatis. Sebagai contoh, ketika anggota meminjam buku pada tanggal 12 Juli 2014, maka sistem secara otomatis memasukkan tanggal 12 Juli 2014 sebagai tanggal peminjaman.
4. Novianto (2013) melakukan evaluasi *maintainability* perangkat lunak berbasis *CodeIgniter Web application framework* dengan sudut pandang pengguna perangkat lunak. Menurut Ilja, Tobias, dan Joost (2007), hasil evaluasi *maintainability* tersebut adalah hasil dengan *lack predictive* (prediksi yang lemah). Permasalahan tersebut dapat diatasi dengan cara menghitung *Maintainability Index* (MI) untuk melakukan evaluasi kualitas *maintainability* berdasarkan sudut pandang subjek *maintenance* perangkat lunak.

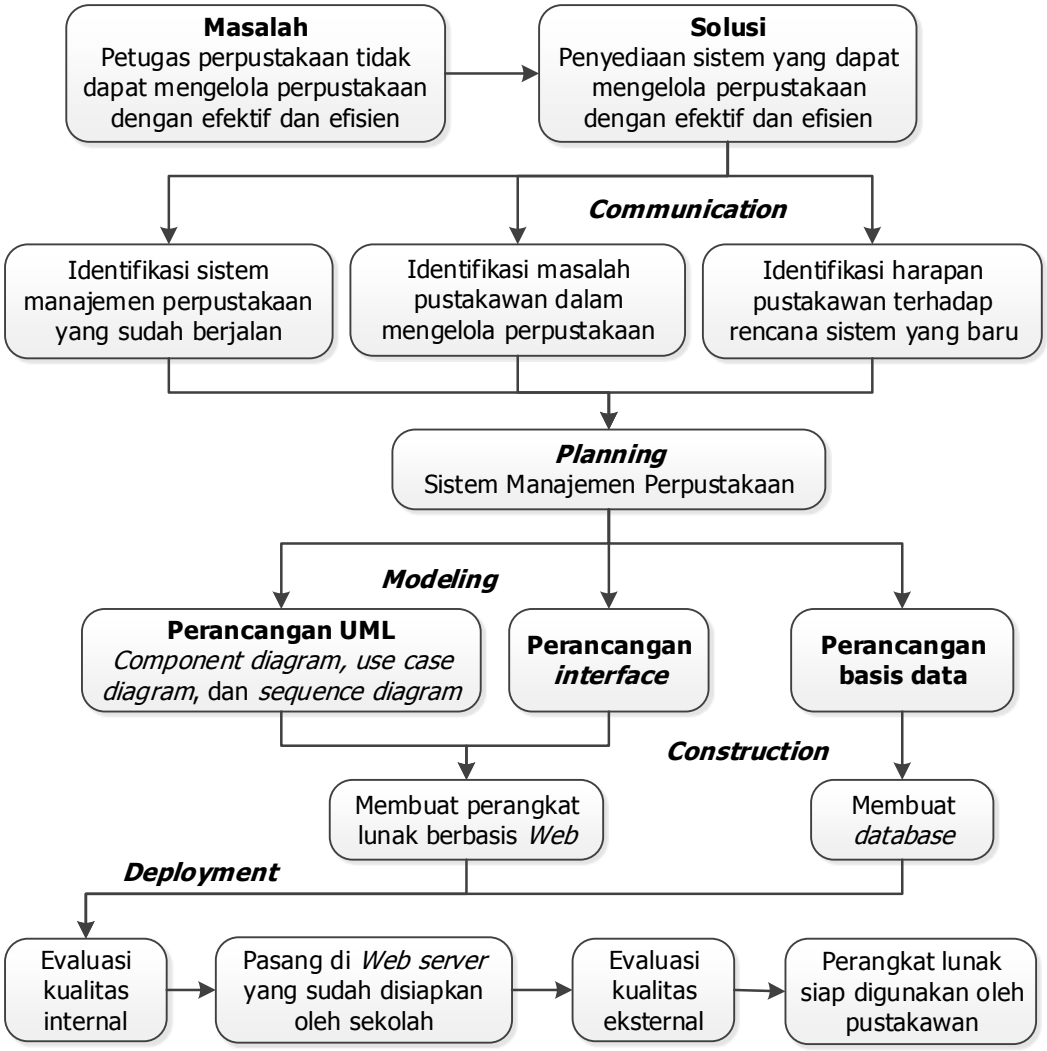
5. Tika (2014) melakukan evaluasi perangkat lunak berbasis *CodeIgniter Web application framework* dengan standar kualitas ISO/IEC 9126. Kelemahan dari penelitian ini adalah Tika mengevaluasi kualitas perangkat lunak hanya pada aspek eksternal (*functionality*, *efficiency*, *reliability*, dan *usability*). Padahal, kualitas internal yang berarti kualitas dari sudut pandang pengembang perangkat lunak dalam memenuhi kebutuhan pengguna juga penting untuk dievaluasi. Permasalahan tersebut dapat diatasi dengan melakukan evaluasi kualitas internal yang terdiri dari *maintainability* dan *portability*.

C. Kerangka Pikir

Permasalahan yang terjadi di SD Muhammadiyah Condongcatur Yogyakarta adalah pustakawan tidak dapat melayani anggota perpustakaan dengan optimal. Hal ini disebabkan belum ada sistem yang dapat mengelola data buku, anggota dan sirkulasi secara efektif dan efisien. Untuk mengatasi hal tersebut, maka diusulkan suatu sistem yang dapat mengatasi berbagai permasalahan pustakawan dalam mengelola perpustakaan.

Langkah awal yang dilakukan dalam penelitian ini adalah komunikasi (*communication*) dengan cara diskusi tanya jawab kepada pustakawan. Hal ini dilakukan untuk mengetahui sistem manajemen yang sudah berjalan, masalah yang dialami pustakawan dalam mengelola perpustakaan, dan harapan pustakawan terhadap sistem baru. Berdasarkan hasil diskusi tersebut, perekayasa menawarkan berbagai solusi pada pustakawan. Langkah yang dilakukan setelah tercapai kesepakatan adalah merancang *Unified Modeling Language* (UML), *interface*, dan basis data. Langkah yang dilakukan setelah selesai merancang adalah membuat *database* dan membangun perangkat lunak.

Sebelum digunakan oleh pustakawan, perangkat lunak yang telah selesai dibangun terlebih dahulu dievaluasi kualitasnya. Kualitas perangkat lunak terdiri dua aspek, yaitu internal dan eksternal. Aspek kualitas yang dievaluasi terlebih dahulu adalah aspek internal. Langkah yang dilakukan setelah perangkat lunak sudah baik secara internal adalah memasang perangkat lunak di *Web server* yang disediakan oleh sekolah untuk dievaluasi kualitasnya dari aspek eksternal. Jika perangkat lunak sudah baik secara eksternal, maka perangkat lunak siap digunakan oleh pustakawan SD Muhammadiyah Condongcatur Yogyakarta.



Gambar 16. Kerangka pikir

D. Pertanyaan Penelitian

Berdasarkan latar belakang pada Bab I, tujuan penelitian pada Bab I, kajian teori pada Bab II, dan kerangka pikir pada Bab II, maka pertanyaan dalam penelitian ini adalah sebagai berikut:

1. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *maintainability*?
2. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *portability*?
3. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *interoperability*?
4. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *suitability*?
5. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *security*?
6. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *usability*?
7. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *time behavior*?
8. Bagaimana kualitas Sistem Manajemen Perpustakaan SD Muhammadiyah Condongcatur Yogyakarta berbasis *Web* dari aspek *maturity*?